USING VIRTUAL PROTOTYPES IN HARDWARE/SOFTWARE CODESIGN

Sebastian Runge[12], Sören Jung[2], Wolfgang Berthin[2], Carsten Wolff [1]

[1]: University of Applied Sciences and Arts Dortmund
Emil-Figge-Str. 42, 44227 Dortmund, Germany
carsten.wolff@fh-dortmund.de

[2] : Intel Mobile Communications GmbH
Düsseldorfer Landstr. 401, 47259 Duisburg, Germany

Keywords: virtual prototype, Hardware/Software Codesign, stream-driven simulation, event-driven simulation, software model, system on chip

Abstract: Hardware/Software Codesign is an approach to combine hardware and software design. It can improve the overall performance and reduce the time to market as well as product costs. A typical architecture, in which Hardware/Software Codesign is used, is the system on chip (SoC). According to the purpose of a SoC it contains specialized components, which can lead to a higher battery capacity or a higher performance, compared to an universal architecture. Typical application areas for SoCs are mobile phone platforms and video processing. A virtual prototype (VP) can be a fully functional software model of a physical hardware like a SoC. They have an early availability in the development cycle and software developers can execute and debug unmodified hardware dependent software on them. They can also be used to identify design errors in hardware. VPs can be implemented in different levels of abstraction. A high abstraction level can have a significant increase of simulation speed. To gain this speed at an early stage of development, stream-driven simulation (SDS) is used. Later when time behavior is of more importance, event-driven simulation (EDS) takes place. To prevent errors and to save the effort of porting SDS to EDS co-simulation of SDS embedded into EDS often is favored. To simulate the microprocessor of the SoC an instruction set simulator is needed which can process the instruction set for a specific microprocessor. This contribution deals with characteristics and benefits of using virtual prototypes in Hardware/Software Codesign for architectures like a SoC.

1. Introduction

The continuous development in digital communication leads to higher data rates for the consumer. This trend results in a rapid growing complexity relating to mobile phone platforms. To compete with this demand it is necessary to use fast and efficient development methodologies.[2]

A design approach which covers the concurrent design of hardware and software is Hardware/Software Codesign. Hardware/Software Codesign can help reduce costs as well as time-to-market, for a system consisting of hardware and software.[1]

With support of virtual prototypes it is possible to execute hardware dependent software on a software model of the hardware, even before the hardware exists. On the one hand design errors of the hardware can be detected earlier on the other hand the integration and verification of the hardware dependent software can start earlier. This leads to a shorter development cycle of the whole product.[3]

2. Hardware/Software Codesign

Hardware/Software Codesign is evolved from two different design approaches which developed widely independent from each other. The hardware part is grown from the design of a material chip to a computer aided design of a digital electronic circuit. The software part has evolved from programming languages very close to the hardware, to hardware independent high level languages. The electronic circuits are usually developed by an electrical engineer or by a hardware developer. The software design is ordinarily done by a computer scientist or a software developer. Both educations are usually independent and take place in a different chair in universities. Even companies have different departments, which software and hardware teams separated from each other.

Both technologies have their advantages. Signal processing and parallel algorithms are usually more efficiently implemented in hardware. Specialized hardware can offer more processing power, or less power consumption. Serial algorithms perform better realized in software running on a microprocessor. The implementation in software can also be done faster and more cost efficient than the development of the same task in hardware. If you design a processor and at the same time the software on that processor, you can make use of the advantages of both technologies. An approach to combine the design of both technologies is Hardware/Software Codesign. [4]

There are several definitions of Hardware/Software Codesign[1][4][5][6]. Schaumont [1] describes Hardware/Software Codesign with the following statement: *"Hardware/Software Codesign is the partitioning and design of an application in terms of fixed and flexible components."*
This statement covers the important aspect of partitioning in Hardware/Software Codesign. Partitioning is the task to decide which components of the system should be designed in hardware (fixed) and which components should be realized in software (flexible). [1] This is an NP-Complete problem and can either be solved manually or automated.[5]

Hardware/Software Codesign can improve overall system performance while reduce the time-to-market as well as product costs. The approach can help a designer to choose between performance and flexibility of a system. An architecture where the approach of Hardware/Software Codesign can take place is the system on chip which is described in chapter 3. [1]

3. System on chip

A system on chip (SoC) is a heterogeneous architecture which combines a microprocessor, several coprocessors, a bus system and memory on a single chip. According to the purpose of the SoC it contains different specialized components which together build a configuration. Such a special configuration is also called a platform. The advantage of a specialized platform compared to a universal processor is founded in its higher processing efficiency. This leads on the one hand to a higher battery capacity and on the other hand to a higher performance of the product. Another advantage is the flexibility to change the configuration which ensures the reusability of the SoC and the hardware dependend software. Typical application areas of SoC are e.g. mobile phone platforms or video processing. Fig. 1 shows a generic template for a system on chip. [1]

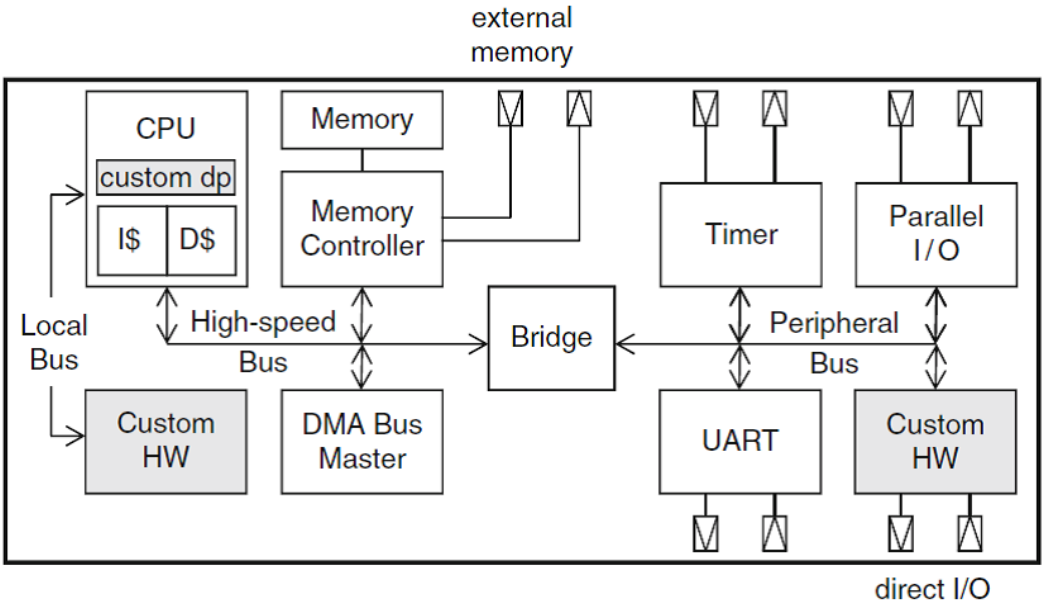

Fig. 1 – Generic template of a system on chip [1]

4. Virtual Prototype for SoCs

Virtual prototypes can be fully functional software models of physical hardware (e.g. SoC). The advantage of virtual prototypes lies in their early availability in the development cycle. This way software developers can begin earlier with development and verification of the hardware dependent software. With virtual prototypes it is possible to execute unmodified hardware dependent software near to the real execution time. Furthermore the developers gain a high level of visibility and control over the system especially on host based virtual prototypes. That is why virtual prototypes are good for debugging hardware dependent software. This visibility and control gain is especially useful on multicore systems. The growing overall complexity of SoCs assures the use of virtual prototyping in Hardware/Software Codesign. Virtual prototypes can also be used to identify possible design errors of the hardware. If there is not enough bandwidth for the firmware you can add e.g. another processor to the hardware model. A hardware error detected early enough can then be corrected in the final hardware. Moreover virtual prototypes are very easy to deploy for many users. [3]

a. Levels of abstraction

Virtual prototypes can be implemented in different levels of abstraction. Corresponding to the level of abstraction, in which the virtual prototype is implemented, a significant rise of simulation speed can be gained. A high abstraction level leads on the one hand to a low detailed implementation, but on the other hand to an increased simulation speed. In general we differentiate between five levels of abstraction based on time granularity (Fig. 2). The first and lowest level of abstraction describes a system in continuous time. In the next level of abstraction discretization takes place. This is the extraction of a discrete subset of events from the system in continuous time. The advantage of the discrete world is the easier mathematical computation. Level three is based on clock cycles. Here we look at what events happen in which clock cycle of the processor. One abstraction level further we look at which processor instruction is processed at what clock cycles. On level five these instructions are summed up into transactions like loading data from the memory. Fig. 2 illustrates these relationships.

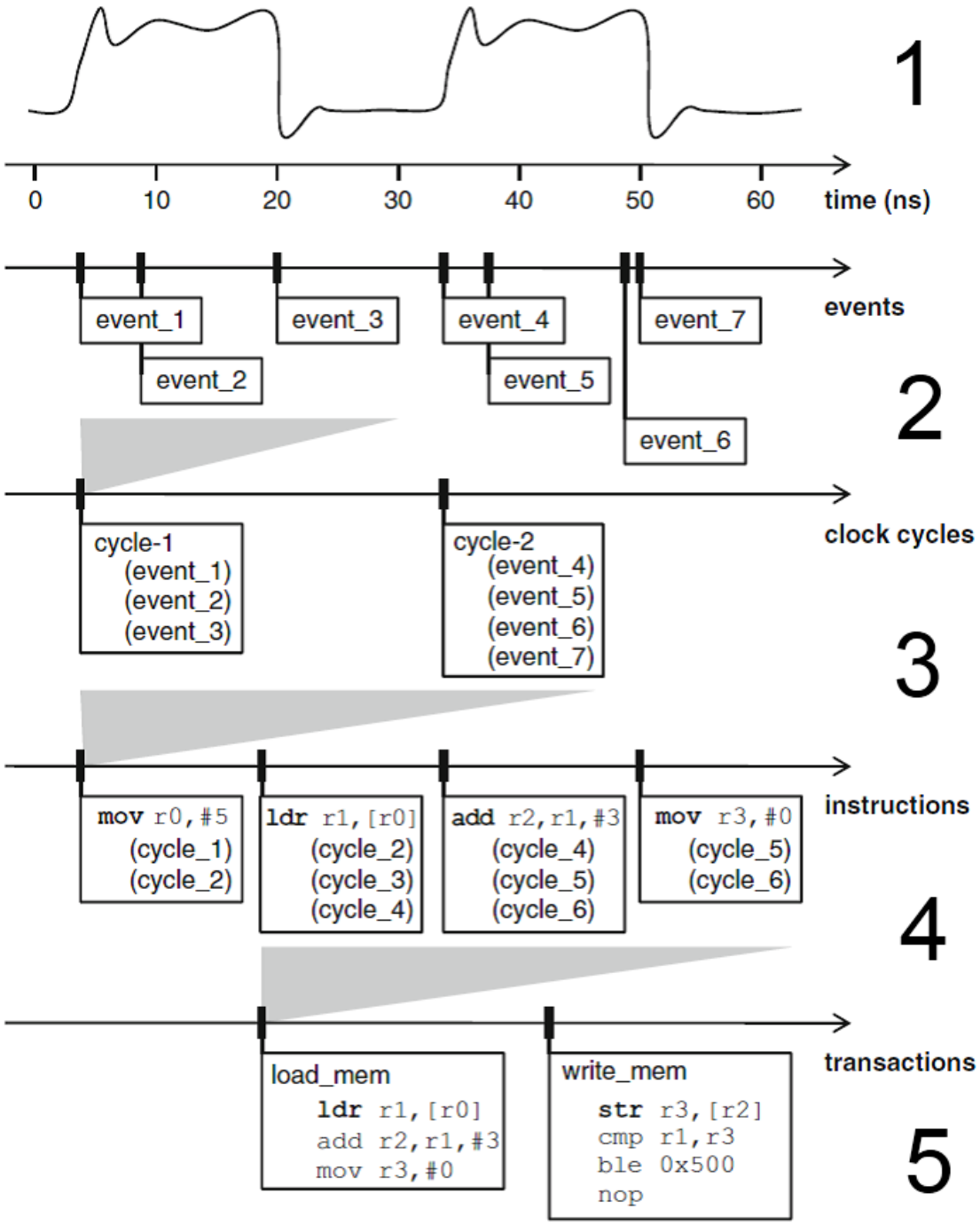

Fig. 2 - abstraction levels [1]

b. Stream-driven simulation

At the beginning of developing a mobile communication platform the focus empirically tends on the development of efficient algorithms. This takes place on a very abstract level where it is important to gain a high simulation speed to concentrate on the development of the algorithms. A favored technique in this phase is stream-driven simulation (SDS). In stream-driven simulations functional blocks communicate over data streams. This takes place free from control flow or time constraints of the hardware. This way the developer can concentrate on the performance of the algorithms.

Fig. 3 shows a typical algorithm model based on stream-driven simulation. In this example the source block (SRC) is a signal generator which represents a base station. The blocks H1-H3 are receiver models which consist of functional blocks (FB_{XY}). The evaluation block (EVL) is an error rate counter which evaluates the quality of the signal. [2]

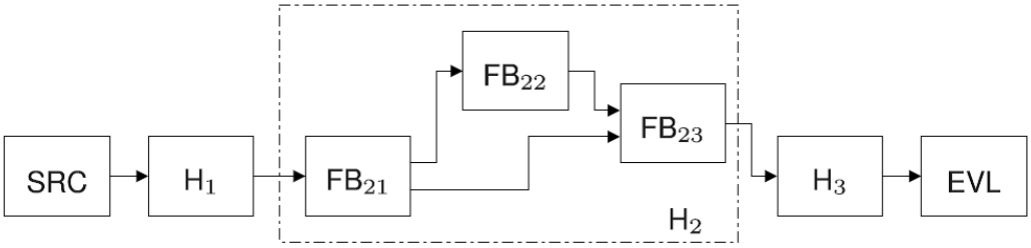

Fig. 3 - stream-driven simulation model [2]

c. Event-driven simulation

Afterwards the event-driven simulation is used to consider the time behavior. Porting a stream-driven model to an event-driven model is time consuming and error-prone. To save this effort and to prevent errors a co-simulation of stream-driven embedded into event-driven simulation often is favored. Fig. 4 shows a co-simulation of stream-driven simulation integrated within event-driven simulation on transaction level (level three). The data exchange between EDS and SDS takes place over buffered ports. To ensure the time constraint on transaction level, the EDS model contains a bus-interface-model (BIM) as well as a state machine (with the states idle (id), load (Ld), process (Pr), report (Rp)) which controls the in- and outputs of the SDS and the execution of the SDS kernel. [2]

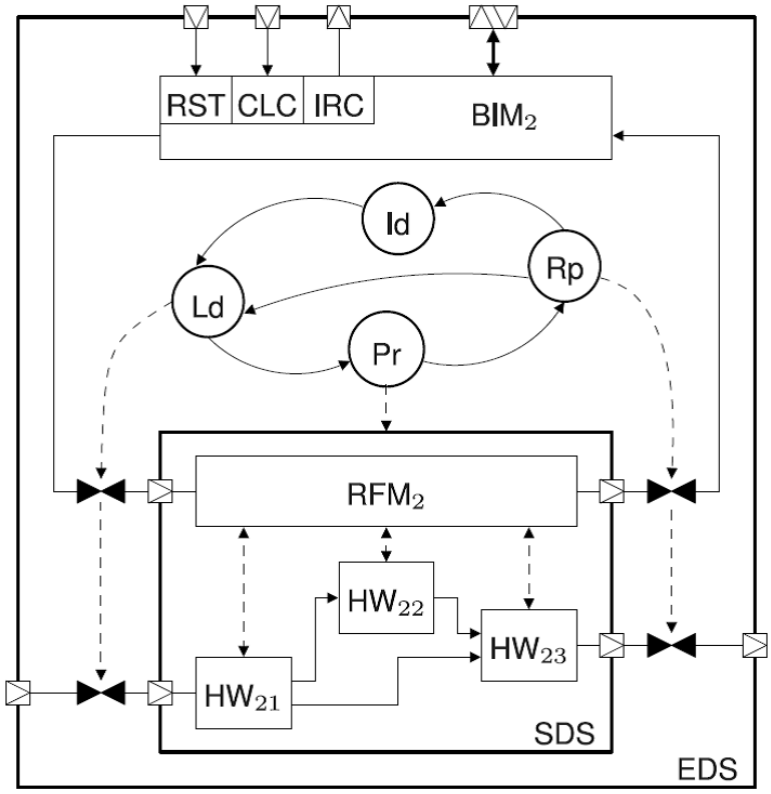

Fig. 4 – transaction level model with event- and stream-driven co-simulation [2]

d. VP model

To bring it all together to a virtual prototype model we also have to consider the microprocessor. To simulate a microprocessor on a PC, simulators are needed, which are able to process a specific instruction set of a certain microprocessor. This is why these simulators are called instruction set simulators (ISS). [1]

Fig. 5 shows the structure of a virtual prototype model which contains event-driven simulation as well as stream-driven simulation embedded into event-driven simulation. To ensure a common time inside the virtual prototype the microcontroller subsystem and the event-driven simulation have to be synchronized. On such a virtual prototype, hardware dependent software can be executed. That is how developing and debugging hardware dependent software can be done before the hardware is available.

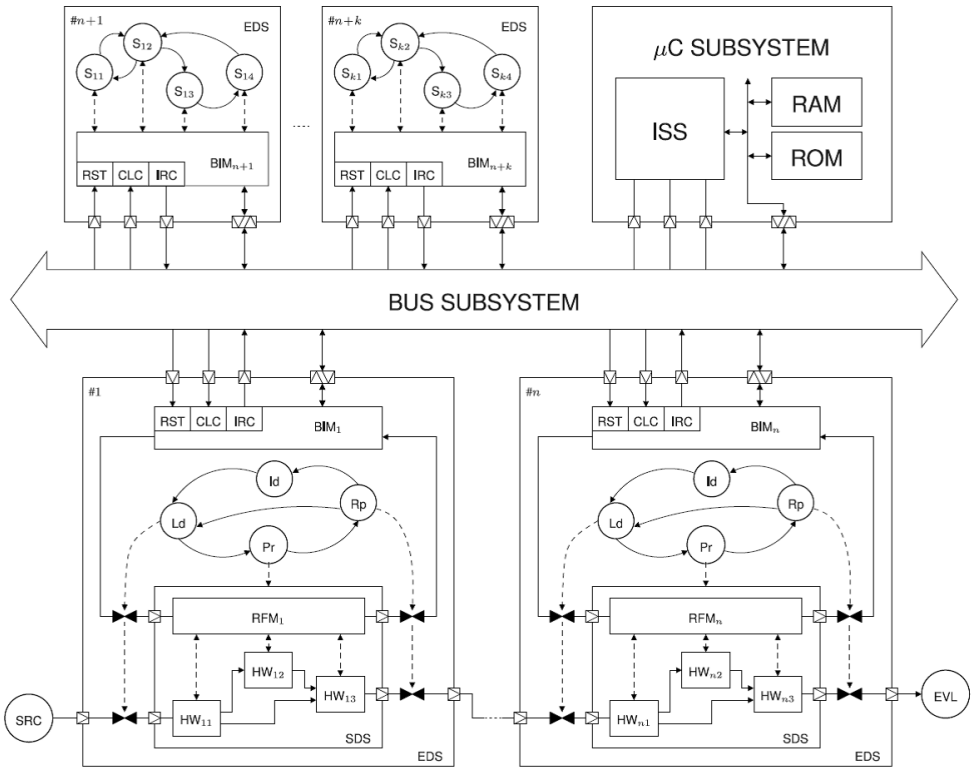

Fig. 5 - Virtual prototype model [2]

5. Results and Conclusion

The aspects described in this paper represent fundamentals of virtual prototypes in Hardware/Software Codesign for architectures like the system on chip. These kind of virtual prototypes were and are used in large digital design projects in semiconductor industry. The "Intel Mobile Communications GmbH" is using virtual prototypes to develop, verify and validate hardware dependent software running on mobile communication platforms. Virtual prototypes can also be used to identify design errors in hardware. The simulation speed can be increased depending on the level of abstraction the VP is implemented in. Stream-driven simulation is used at an early stage of development where simulation speed to develop algorithms is absolutely necessary. To consider time behavior, event-driven simulation is used in a later phase. To avoid porting the stream-driven simulation, co-simulation of both often takes place. To simulate a microprocessor an instruction set simulator for that specific microprocessor is needed. This VP-Model used in Hardware/Software Codesign can help to improve the quality of the product and to reduce time to market as well as product costs.

6. References

[1] Schaumont PR. A Practical Introduction to Hardware/Software Codesign. Boston, MA: Springer Science+Business Media LLC; 2010.

[2] Ecker W, Müller W, Domer R. Hardware-dependent software: Principles and practice. Berlin: Springer; 2009.

[3] Amos D, Lesea A, Richter R. FPGA-based prototyping methodology manual: Best practices in design-for-prototyping. Mountain View, Calif: Synopsys; 2011.

[4] R. Gessler and T. W. Mahr, Hardware-Software-Codesign: Entwicklung flexibler Mikroprozessor-FPGA-Hochleistungssysteme, 1st ed. Wiesbaden: Vieweg, 2007.

[5] M. Meerwein, Ein wiederverwendungsorientiertes Hardware-Software-Codesign-System für Mikrocontroller-basierte Systeme. Düsseldorf: VDI-Verl, 2002.

[6] R. Niemann, Hardware software co-design for data flow dominated embedded systems, 1998.